# Đề: cho một đồ thị vô hướng hãy kiểm tra xem đồ thị đó có phải là một cái cây hay không.

## Ý tưởng:

* Có nhiều điều kiện để một đồ thị là một cái cây nhưng điều kiện để làm bài này là: Đồ thị à môt cây nếu đồ thị đó có số cạnh = số đỉnh-1 và đó là đồ thị liên thông.
* Chứng minh đồ thị liên thông thì chúng ta sẽ dùng DFS hoặc BFS nhưng bài này ta dùng DFS co đệ quy

## Mã giả:

* DFS(int u)
* thăm node đầu tiên
* tiêp tục xét các node kề với nó.
* Nếu chưa thăm thì thăm và DFS(v)
* Boolean CheckTree()
* Nếu số cạnh!= đỉnh -1; return false;
* DFS(1);
* Nếu có node chưa thăm return false;
* Retrun true;

## Độ phức tạp

O(n+m) với n là số cạnh, m là số đỉnh

## Thuật Toán

#include <iostream>

#include <vector>

using namespace std;

const int MAXN = 1000;

vector<int> adj[MAXN];

bool visited[MAXN];

int n, m; // n: số đỉnh, m: số cạnh

void dfs(int u) {

    visited[u] = true;

    for (int v : adj[u]) {

        if (!visited[v]) dfs(v);

    }

}

bool isTree() {

    // Điều kiện 1: số cạnh = n - 1

    if (m != n - 1) return false;

    // Điều kiện 2: đồ thị phải liên thông

    dfs(1); // bắt đầu duyệt từ đỉnh 1

    for (int i = 1; i <= n; i++) {

        if (!visited[i]) return false; // có đỉnh chưa thăm -> không liên thông

    }

    return true;

}

int main() {

    cout << "Nhap so dinh va so canh: ";

    cin >> n >> m;

    cout << "Nhap cac canh (u v):\n";

    for (int i = 0; i < m; i++) {

        int u, v;

        cin >> u >> v;

        adj[u].push\_back(v);

        adj[v].push\_back(u);

    }

    if (isTree()) cout << "Do thi la mot cay.\n";

    else cout << "Do thi khong phai la cay.\n";

    return 0;

}

## Input

5 4

1 2

1 3

3 4

3 5

## Output

